Amendments to the Specification:

Please replace paragraph [0007] with the following replacement paragraph [0007]:

[0007] The present invention is directed to systems and methods for compression of a b-tree using normalized index keys. A b-tree may contain multiple keys, and each key may contain several columns of different types. A key may be viewed as a concatenation of the binary representation of the individual column values. However, each type may have its own particular binary representation, making it difficult to compare values of different types. A normalized key may be compared with another normalized key without any instantiation of types or use of any type specific functions.

Please replace paragraph [0020] with the following replacement paragraph [0020]:

[0020] At step 111, a key is selected for normalization. The key can consist of multiple columns of multiple types. A variable (e.g., "max") is set to the total number of columns in the key. A counting variable (e.g., "count") is set to zero, and is incremented by one each time one of the columns in the key is normalized. It may be appreciated that by comparing the values of max and count in can be determined if there are any remaining columns of the key to normalize. A variable (e.g., "normalizedkey") is also created to store the value of the normalized key as it is generated.

Please replace paragraph [0023] with the following replacement paragraph [0023]:

[0023] For example, bits 0 through 3 of the marker byte can be set to the hexadecimal value 0x05. This value is the same for all marker bytes. Bit 4 is set to 0 if ~~signifying that~~ null values are sorted high, and to 1 if null ~~nulls~~ values are sorted low. Sorting null values high or low is a property of the underlying b-tree. Bit 5 may be relevant only where a column value is of type bit. If the type of the column is not bit, bit 5 is set to 0. If the type of the column is bit, then bit 5 is set equal to the column value. Bit 6 is set to 1 if the column value is null, or to 0 if the column value is not null. However, if bit ~~Bit~~ 4 is set to 1, indicating nulls sorted low, the resulting bit ~~Bit~~ 6 value is inverted. Bit 7 is set to 0 if the underlying b-tree is sorted in ascending sort order, and is set to 1 if the underlying b-tree is sorted in descending sort order. The resulting marker byte is now appended to normalizedkey.

Please replace paragraph [0062] with the following replacement paragraph [0062]:

[0062] At step 501, a memory page compressing event has been triggered, such as a page split. It may be desirable for the b-tree compression to be transparent to the end user and to perform the compression of the b-tree at some time when the operation will be the least intrusive. To this end, the present embodiment desirably compresses keys in a given memory page right before a slow operation such as a page split. A page split is a costly operation that occurs when a page of memory has been filled with keys. The memory page is split into two separate memory pages. It may be appreciated that by compressing before the page split, any cost associated with the compression desirably will be offset by the savings incurred by not splitting the memory page. The memory page can consist of compressed and uncompressed keys, because it is desirable to insert uncompressed keys and only compress keys when approaching a page split.